PROBABILITY QUANTIZATION FOR MULTIPLICATION-FREE BINARY ARITHMETIC CODING *

Kar-Ming Cheung
Communications Systems Research
238-420
Jet Propulsion Laboratory
4800 Oak Grove Drive
Pasadena, CA 91109

EXTENDED ABSTRACT:

The binary arithmetic code is a crucial element of many practical stlatc-of-the-art lossless and lossy compression schemes. The basic idea behind binary arithmetic coders is to encode a sequence of O's and 1's into a codestring that represents a binary fractional value. This value is obtained by successively dividing a base interval into two parts according to some estimate of the symbol probabilities, and then select ing the interval that corresponds to the symbol sent .

The key to an efficient implementation of the binary arithmetic coding procedure is to avoid performing the time- consuming multipication and division operations in the probability update for each binary symbol sent. 1 BM's QM-coder [1] keeps track of two fixed-lcngtb registers A and C, where A represents the size of the current interval, and C indicates the base of the current interval. By means of a normalization process A and C are kept within a specific range. By a simple approximation that requires A to be in the range of $0.75 < A < 1.5$, the QM-coder replaces multiplications with simple additions arid subtractions. Witten ct. al. proposed a more intuitive a] proach to perform binary arithmetic coding [2]. Witten's binary arithmetic coding procedure keeps track of two values *high* and *low*, where *high* and *low* correspond to t he top and t he bottom of the current interval. Witten suggested to constrain the potability of the less probable symbol to the nearest integral power of $\frac{1}{2}$, so that multiplications can be replaced by simple shifts. Using a binary entropy argument, Witten tabulated the optimal probability ranges for each power of $\frac{1}{2}$, and showed that the womt-case efficiency is about 95.0% (Figure 1).

In this article wc improve upon Witten's results by approximating the probability of the less probable symbol with a fraction of the form 2^{-l} or $2^{-l} + 2^{-l-2}$ for $l = 1,2, \ldots$. It is easy to show that multiplying a number by $2^{-l-1} + 2^{-l-2}$ is equivalent to right-shifting it by $l + 0.415$ bits. Computationally this correponds to replacing a multiplication operation with 2 shifts and an add. As wc will show later, this scheme improves the worst-case coding efficiency to 98.5%.

The following is a sketch on how to optimally quantize the probability of the less probable symbol to achieve the aforementioned computational efficiency. Wc usc a similar

* The research described in this paper was carried out by Jet Propulsion Laboratory, (California Institute of Technology, under a contract with National Aeronautics and Space Administration

approach as Witten's. Let p, $0 < p \leq 0.5$, be the true probability of the less probable symbol. The question is to choose a step-wise probability quantization function Q(p) of p such that the average code length per symbol, namely $p \log_2 (Q(p)) - (1-p) \log_2 (1 - Q(P))$, *is minimized.* The *design* of *Q(p) is* complexity-driven, not performance-driven. However as we will show later, that we do not sacrifice! much by quantizing p into the form 2–Z or $2^{-l-1} + 2^{-l-2}$ for $l = 1, 2, \ldots$. We examine two different cases.

Case 1: $2^{-l-1} + 2^{-l-2} < p \leq 2^{-l}$

This *corresponds* to finding the breakpoint p' such that

$$p'(l + 0.41504) - (1 - p') \log_2 (1 - 2^{-l-1} - 2^{-l-2}) = p' l - (1 - p') \log_2 (1 - 2^{-l}).$$

Case 2: $2^{-l-1} < p \leq 2^{-l-1} + 2^{-l-2}$

This corresponds to finding the breakpoint p' such that

$$p'(l+1) - (1-p') \log_2 (1 - 2^{-l-1}) = p'(l + 0.41504) - (1 - p') \log_2 (1 - 2^{-l-1} - 2^{-l-2}).$$

We use the same performance efficiency definition as Witten 's, which is given by the entropy as a fraction of the aver-age code length,

$$\text{efficiency} = \frac{-p \log_2 p - (1-p) \log_2 (1-p)}{pQ(p) - (1-p) \log_2 (1 - 2^{-Q(p)})}.$$

We tabulated the optimal probability range and the wcnst-case efficiency for each quantized probability value (Figure 2).

We implemented a *heterogeneous* binary arithmetic coder [3] using this probability quantization scheme. The heterogeneous binary arithmetic coder is multiplication-free, and it uses only byte-wise operations. Preliminary results show that it gives better compression performance than the QM-coder at both high-entropy range and low-entropy range.

References:

[1] W. Pennebaker, and J. Mitchell, *JPEG: Still Image Data Compression Standard,* Van Nostrand Reinhold, New York, 1993.

[2] 1. Witten, R. Neal, and J. Cleary, *Communications of the ACM,* June 1987 Volume 30, number G.

[3] K. Cheung, "Coarse-Grain Binary Arithmetic Coding", to appear.

Proability Range	Right-Shift	Effective Probability	Efficiency
0,36907-0.50000	1	**0 . 5**	**0.950**
0,18193-0.36907	2	**0.25**	**0.973**
0.09053-0.18193	3	**0.125**	**0.981**
0.04517-0.09053	4	0.0625	0.985
0.02256-0.04517	5	0.03125	0.987
0.01128-0.02256	6	0.015625	0.989
0.00564-0.01128	7	0.0078125	**0.990**
0.00282-0.00564	8	**0.00390625**	**0.991**

Figure 1 : Witten's Probability Quantization Scheme

Proability Range	Right-Shift	Effective Probability	Efficiency
0.43683-0.50000	1	**0 . 5**	**0.988**
0.31018-0,43683	1.415	**0.375**	**0.985**
0.21767-0.31018	2	**0.25**	**0.994**
0.15453-0.21767	2.415	**0.1875**	**0.991**
0.10872-0.15453	3	**0.125**	**0.996**
0,07716-0.10872	3.415	**0.09375**	**0.994**
0.05433-0.07716	4	**0.0625**	**0.997**
0.03856-0.05433	4.415	**0.046875**	**0.995**
0.02716-0.03856	5	**0.03125**	**0.998**
0.01927-0.02716	5.415	**0.0234375**	**0.996**
0.01358-0.01927	6	**0.015625**	**0.998**
0.00964-0.01358	6.415	**0.01171875**	**0.996**
0.00679-0.00964	7	**0.0078125**	**0.998**
0.00482-0.00679	7.415	**0.005859375**	**0.997**
0.00339-0.00482	8	**0.00390625**	**0.999**

Figure 2 : improved Probability Quantization Scheme